VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

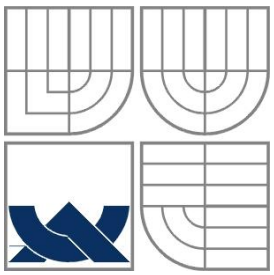
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZPOZNÁVÁNÍ ČÍSLIC POMOCÍ NEURONOVÉ SÍTĚ

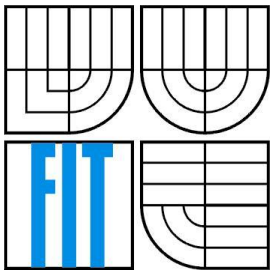
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE ZDENĚK DOUPOVEC
AUTHOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZPOZNÁVÁNÍ ČÍSLIC POMOCÍ NEURONOVÉ SÍTĚ

NEURAL NETWORK NUMBER RECOGNITION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ZDENĚK DOUPOVEC

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JANA ŠILHAVÁ

BRNO 2009

Abstrakt

Tato práce popisuje základními pojmy a principy v oboru neuronových sítí. Blíže se pak věnuje problematice vícevrstvých perceptronových sítí, konkrétně metodě back-propagation. Jsou zde rozebrány výhody a nevýhody zmíněné metody, návrh možného systému rozpoznávání číslic pomocí back-propagation. Cílem je získat konkrétní výsledky z programu schopného rozpoznávat čísla.

Abstract

This work describes the basic concepts and principles in the field of neural networks. Closer then deals with the problem of multilayer perceptron networks, namely back-propagation method. There are analyzed the advantages and disadvantages of these methods, the proposal of possible digits recognition system using back-propagation. The aim is to obtain concrete results from the program whitch is able to recognize numbers.

Klíčová slova

Neuron, neuronová síť, perceptron, back-propagation, rozpoznávání číslic, učení, rozpoznávání.

Keywords

Neuron, neural network, perceptron, back-propagation, number recognition, learning, recognition.

Citace

Zdeněk Doupovec: Rozpoznávání číslic pomocí neuronové sítě, bakalářská práce, Brno, FIT VUT v Brně, 2009

Rozpoznávání číslic pomocí neuronové sítě

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jany Šilhavé. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Zdeněk Doupovec
10. Května 2009

Poděkování

Chtěl bych poděkovat své vedoucí Ing. Janě Šilhavé za odborné vedení, za ochotu poskytnout mi informace a rady, a za čas, který mi věnovala. Dále pak kamarádu Tomáši Janíkovi a své rodině za podporu.

Obsah

1 Úvod

V současné době se problému neuronových sítí věnuje nespočet vědců. Uvědomujeme si možný potenciál, který v sobě skrývají. Můžeme směle tvrdit, že neuronové sítě se využívají téměř všude a s postupem času doufejme, že se tento trend bude dále rozšiřovat. Nezastupitelné uplatnění nalezneme v lingvistice, vědě o neuronech, řízení procesů, přírodních a společenských vědách, kde se pomocí nich modelují nejen procesy učení a adaptace, ale i široké spektrum různých problémů klasifikace objektů a také problémů řízení složitých průmyslových systémů. Při vytváření modelů umělých neuronových sítí nám nejde o vytvoření identických kopií lidského mozku, ale napodobujeme pouze některé jeho základní funkce.

Tato práce má za úkol vyzkoušet a testovat neuronové sítě k řešení konkrétních problémů, ověřit funkční metody a principy, případně navrhnout modely, které by stávající dále rozšiřovaly. Chci na neuronové sítě nahlížet jako na neoddělitelný celek mnoha oborů, počítačové grafiky, umělé inteligence, zpracování obrazu a videa apod., a pokusit se tento fakt zohlednit i ve své práci.

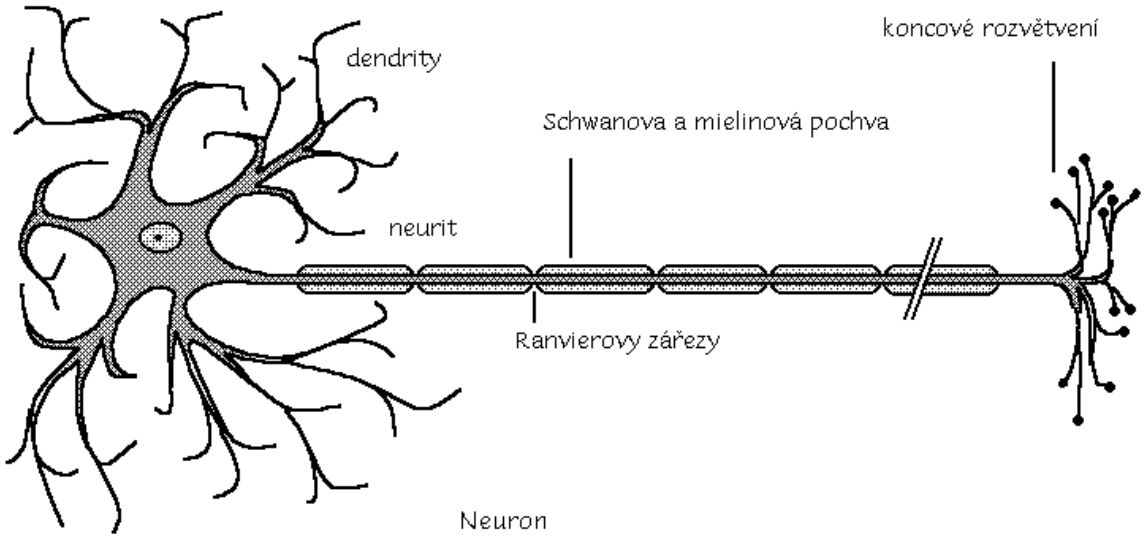
V kapitole [2] seznamuje čtenáře s modelem biologického neuronu, který vytvořila příroda, protože je považován za základní kámen celého oboru. Nalezneme zde matematický model a vůbec popis biologického neuron a neuronové sítě tak, jak jej vytvořil člověk. Bližší seznámení s konkrétní neuronovou sítí (metoda back-propagation) používanou pro rozpoznávání číslic pak v podkapitole [2.5]. Další částí je návrh možné implementace [3]. Zde narazíme na konkrétní problémy, jež se mohou při vytváření systému objevit, metody jako prahování, vyhledávání objektů v obraze apod., návrh systému pro učení [3.1] a rozpoznávání[3.2]. Poslední částí [4] je pak samotná implementace a porovnání dosažených výsledků. Závěrem celé práce [5] je hodnocení, návrhy na další možnosti rozšiřování a poslední slovo autora.

2 Neuron a neuronové sítě

V této kapitole popisuji biologický neuron a jeho neživou reprezentaci používanou v informačních technologiích. Vysvětluji principy fungování neuronů, nechybí zde ani jejich srovnání. Podrobněji je zde popsána metoda back-propagation, kterou jsem si zvolil jako vhodnou pro rozpoznávání.

2.1 Biologický model neuronu

Lidský mozek se skládá z asi 10^{12} bohatě rozvětvených nervových buněk, neuronů, vedoucích vzruchy. Ty spolu komunikují prostřednictvím sítě vazeb, vstupů a výstupů a pomocí elektrických výstupů v chemickém prostředí. Neurony jsou základními stavebními prvky nervové soustavy, které se zaměřují na sbírání, uchovávání, zpracování a přenos informací. Existuje celá řada různých neuronů lišících se ve tvaru a velikosti. Většina má ale shodné části s typickým míšním motoneuronem (viz Obr. 2.1) [8].



Obr. 2.1 Biologický model neuronu

Neuron se skládá z několika hlavních částí. Tělo, z něhož vychází jeden výstup (*axon*) a množství *dendritů*, které tvoří vstup neuronu. Axon může být velmi dlouhý, i několik metrů u velkých zvířat. U člověka je nejdelší axon asi 1 m dlouhý. Vede od páteře až po konečky prstů na nohou. Axon je chráněn *Myelinovou pochvou*, tvořenou *Schwannovými buňkami*. Ta není souvislá, vytváří asi 1mm dlouhé segmenty (internodia) členěné *Ranvierovými zářezy*. V místech zářezu je na membráně axonu velké množství elektricky řízených iontových kanálů. Výplň mezi neurony tvoří drobné, bohatě rozvětvené buňky *neuroglie* (glie). Tyto buňky vzruchy nevedou, vyživují neurony a odvádějí odpadní látky metabolizmu v nervové tkáni.

Jeden neuron má v lidském mozku spojení v průměru asi 10000-100000 spojů s jinými neurony. Zajímavostí je, že pokud vypadnou některé z neuronů dodávajících informace, tak se výsledné chování neuronové sítě nezmění.

Dendrity se s axony sousedních neuronů stýkají prostřednictvím *synapse*. Přenášené signály jsou elektrické impulsy, jejichž přenos je ovlivněn uvolňováním chemických látek v synapsích. Synapse působí nejen jako rozhraní mezi jednotlivými neurony, ale také přispívají na vytváření paměťových stop. Neuron má svou vnitřní aktivitu, která je závislá na jeho historii (např. dosavadním průběhu aktivity) a na dráždění ostatními neurony nebo receptory. Aktivace neuronu nastane tehdy, překročí-li hodnota budících vstupních signálů hodnotu tlumících signálů o určitou prahovou hodnotu. V praxi situace vypadá tak, že neuron přijímá signály (podněty) ze sousedních neuronů. Podnět může být elektrický, chemický nebo mechanický. Teprve až nashromážděné hodnoty překročí prahovou, pošle po neuritu svůj signál (toto je označováno jako pálení neuronu). Tento signál můžeme registrovat jako elektrický děj. Rychlost signálu (vzruchu) závisí na síle myelinové vrstvy (ta izoluje neurit od ostatních neuronů) a pohybuje se od 0.5 m/s do 120 m/s. Když vzruch dojde k synapsi, elektrický signál se mění na chemický. V zakončení neuritu jsou váčky obsahující *mediátor*. Ten se uvolní do synaptické štěrbiny. Na druhé straně synapse je mediátor zaznamenán chemoreceptory a vzruch je opět převeden na elektrický signál. Tímto způsobem se síří signál po celé neuronové síti [2].

2.2 Umělý model neuronu

K reprezentaci biologického neuronu se používá formální neuron. Často se v literatuře setkáváme s pojmem perceptron. Jde o formální neuron použitý ke konkrétním příkladům. Formální neuron (dále jen neuron) je umělý zjednodušený model biologického neuronu, od kterého očekáváme stejné (nebo alespoň podobné) chování. Neuron se skládá z několika vstupů a právě jednoho výstupu. Každému vstupu x_i přiřazujeme určitou váhu w_i. Neuron má jeden trvalý vstup o váze w_0, který je se nazývá práh.

Pro vyjádření vztahu vstupních a výstupních hodnot neuronu se používá následující vzorec [7]:

$$y = f\left(\sum_{i=1}^{N} w_i\, x_i + \Theta\right) \tag{2.1}$$

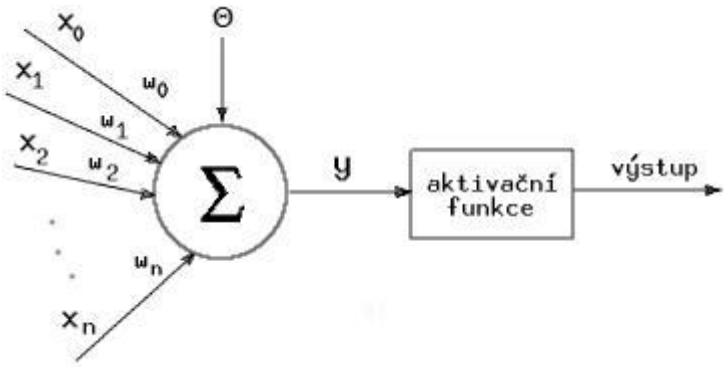
x_i Vstupy neuronu, počet vstupů je N

w_i Váhové vstupy neuronu (také označované jako synaptické váhy), počet je N

Θ Práh neuronu

f Aktivační funkce, přenosová funkce. Argument je označován jako vnitřní potenciál neuronu

y Výstup neuronu

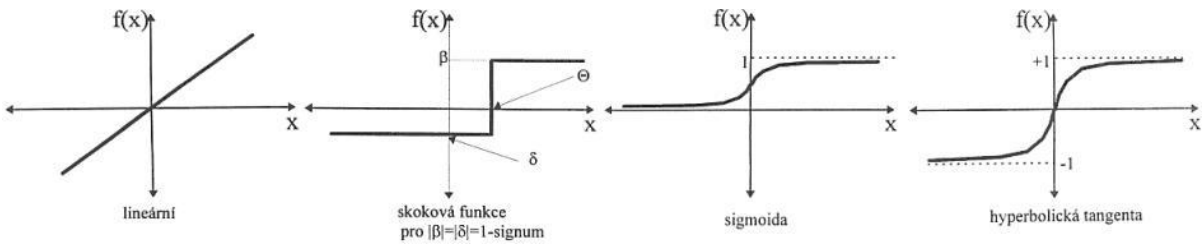


Obr. 2.2 Formální model neuronu

Výstup neuronu je určen funkční hodnotou aktivační funkce vnitřního potenciálu neuronu. Podle toho, jakou aktivační funkci použijeme, dostáváme jiné výstupy a tedy jiné neurony. Základní typy aktivačních funkcí jsou:

- Lineární
- Skoková
- Sigmoida
- Hyperbolický tangens

Na obrázku č. 2.3 [3] jsou znázorněny průběhy jednotlivých funkcí. Z hlediska neuronů a neuronových sítí jsou nejzajímavější sigmoida a hyperbolický tangens. Obě jsou rostoucí a liší se jen oborem hodnot. Sigmoida je definovaná na intervalu (0,1) a hyperbolický tangens na intervalu (-1,1).



Obr. 2.3 Aktivační funkce

Za pozornost stojí rovnice zmiňovaných funkcí sigmoidy a hyperbolické tangenty [10]. Zde totiž hodnota λ určuje „roztažení“ funkce po ose x. Tato vlastnost nám dovoluje vkládat do funkce větší hodnoty.

$$P(t) = \frac{1}{1 + e^{-\lambda t}} \qquad Q(t) = \frac{2}{1 + e^{-\lambda t}} - 1$$

(2.2)

Hlavní důvod používání právě těchto funkcí je ale možnost je jednoduše derivovat. Její derivace se nikde neblíží nekonečnu [4].

$$P(t)' = P(t)(1 - P(t)) \qquad Q(t)' = 1 - Q(t)^2 \tag{2.3}$$

2.3 Srovnání modelů

Při porovnávání obou modelů lze narazit na části, které se chovají naprosto stejně, což je zřejmé, pro tento účel byl model neuronu vytvořen. Nyní konkrétněji.

Vstupní signály přicházející po dendritech odpovídají vstupům u formálního neuronu. V případech obou neuronů existuje mezní hodnota (ekvivalent prahu), která neuron aktivuje. Aktivační funkci bychom mohli přirovnat k mechanismu, který v živém neuronu zabezpečí aktivaci pouze, je-li celkový potenciál větší než asi 15mV [2]. Výstup formálního neuronu odpovídá axonu.

Je nepopiratelné, že biologický model je dokonalejší. Pokud vytvoříme neuronovou síť, musíme zohledňovat fakt, že počítače ještě stále pracují sekvenčně a nejsou schopny vypočítávat souběžně několik operací. I u více-jádrových procesorů dochází k sekvenčnímu zpracování a k nutné synchronizaci.

2.3.1 Modely neuronových sítí

Vzhledem k sekvenčnímu zpracovávání dat procesorem, bylo nutné přistupovat takto i k neuronovým sítím. Byly vytvořeny modely, které maximálně zohledňují zmíněnou problematiku.

Příklady základních modelů:

- *model Sutton-Barto, model Hebb* – Tyto modely považují statický pohled na problematiku za nedostatečný. Jsou inspirovány Darwinovskou představou sobeckého chování základního elementu, který se snaží maximálně využít své vstupy podle svého měřítka hodnot. Byly navrženy tak, aby pro dostatečný počet neuronů byly schopny ovlivňovat vnitřní parametry, respektive že se neurony mohou upravit sami podle opakovaných vstupních podnětů a stát se stabilní. Učení nastává tehdy, když jsou neurony stabilizovány.

- *model Hopfield, model Bolzmann* - Topologie sítí je naprosto homogenní, každý neuron je spojen se všemi ostatními, spojení jsou symetrická. Hopfieldova síť používá dvě různé binární prahové jednotky (-1,1 a 0,1), takže existuji dvě možnosti její aktivace. Je vhodná pro data s binární reprezentací např. černo-bílé obrazy. Často se používá jako asociativní paměť nebo pro řešení optimalizačních problémů. Síť se velice rychle a snadno adaptuje – vytváří si

tzv. stabilní stavy podle tréninkových vzorů. S jistotou konverguje k lokálnímu minimu, ale nezaručuje konvergenci k jednomu z uložených vzorů. Bolzmanův stroj (model) definuje síť jako soustavu stochastických částic, které se snaží dosáhnout stavu s nejmenší energií. V tomto (definicí částic) se od Hopfieldova modelu liší. Jinak jsou si ale oba modely velmi podobné, oba uvažují jednotky s energií, výpočetní vzorce pro celkovou energii jsou stejné.

- *Hammingův model* - Tato síť je nejjednodušším příkladem kompetičního modelu. Ten se skládá se ze dvou vrstev neuronů: z vrstvy receptorů a z vrstvy vstupních neuronů (selfish neurony). Výstupní neurony nemají fiktivní vstup. Výstup každého vstupního neuronu je propojen se všemi výstupními neurony. Jedná se o variantu s učitelem a binárním vstupem. Lze ji také popsat jako třídič dle nejmenší chyby: k danému vstupu najde kategorii, jejíž reprezentant má od vstupu nejmenší tzv. Hammingovu vzdálenost (tj počet odlišných vstupů). První vrstva sítě počítá Hammingovu vzdálenost (respektive doplněk), druhá laterální inhibicí vybírá maximum (správnou kategorii).

- *Kohonenův model* - Kohonenova samo-organizující se síť je typ sítě, které při učení nepotřebují učitele. Je založena na algoritmu shlukové analýzy, tj. na schopnosti nalézt určité navzájem závislé vlastnosti přímo v překládaných trénovacích datech bez přítomnosti nějaké vnější informace. Algoritmus vytváří nízko-dimenzionální (obvykle dvou-dimenzionální) rozlišitelné reprezentace vstupních tréninkových sad, které se nazývají mapy. Každý neuron ve výstupní vrstvě je výstupem a jejich počet (výstupů) je tedy roven počtu neuronů. Učení v Kohonenově síti probíhá tak, že se učící algoritmus snaží uspořádat neurony v mřížce do určitých oblastí tak, aby byly schopny klasifikovat předložená vstupní data.

- *Feed-forward model* – nerekurentní, nejjednodušší a vůbec první model neuronové sítě. Informace se pohybuje pouze jedním směrem, od vstupní vrstvy přes vrstvu skrytých neuronů na vrstvu výstupní. Síť neobsahuje žádné smyčky ani cykly. Podrobněji se této metodě, a její konkrétní variantě back-propagation, budu věnovat v následujících kapitolách.

2.4 Perceptronová síť

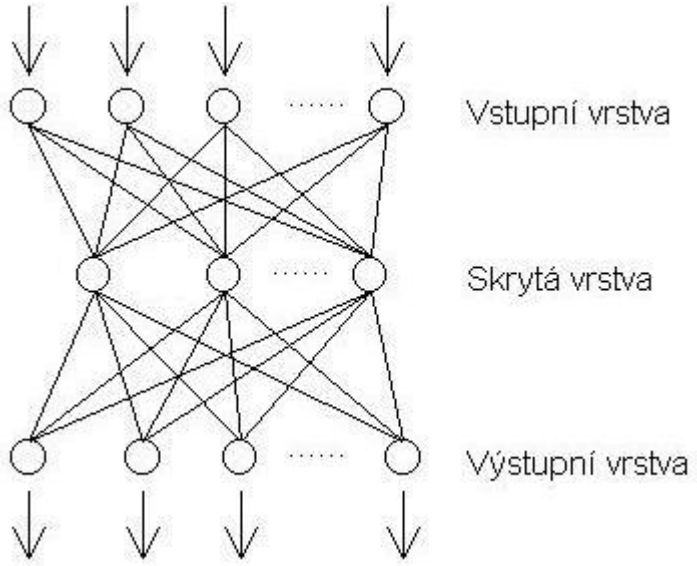
Je síť, která má za vzor biologickou neuronovou síť. Skládá se z perceptronů, které jako celek používají paralelní zpracování informace. Používají se jak k řešení problémů v oboru umělé inteligence, tak k simulaci biologické neuronové sítě. To je velmi výhodné, není totiž potřeba vytvářet biologický model.

Dále budu používat pojem perceptron namísto neuronu. I když pojem vznikl později, vyjadřuje totéž.

Zjednodušeně si lze fungování perceptronové sítě představit jako funkci, která se snaží pro nějaký vektor vstupních hodnot přiblížit vektoru výstupnímu tak, že nastavuje hodnoty vektoru vah. Běžně se používá základní model model se třemi vrstvami:

- Vrstva vstupních perceptronů
- Vrstva skrytých perceptronů
- Vrstva výstupních perceptronů

Vrstva vstupních a výstupních perceptronů slouží jako rozhraní, snažící se požadované vstupní a výstupní informace předat skryté vrstvě ke zpracování.



Obr. 2.4 Perceptronová síť [5]

Protože se při předávání a porovnávání informací s výstupní vrstvou uplatňuje proces změny vah neuronů, říkáme, že se perceptronova síť *adaptuje* (učí). Hodnoty vah zde reprezentují paměť perceptronové sítě. Po adaptaci nastává *vybavování* (fáze aktivní). Spočívá ve "vypočítání" výstupu ze vstupních informací, přičemž síť zohledňuje všechny naučené vzory.

2.4.1 Učení

Učení se v perceptronové síti, realizuje nastavováním vah mezi jednotlivými uzly. Vahám se přisoudí počáteční hodnoty, které mohou být náhodně zvolené, nebo vybrané podle nějakého podobného případu. Poté se na vstupní vektor přivede trénovací množina. Síť následně poskytne patřičný výstup. Dle typu porovnání získaných výstupních hodnot rozlišujeme dva typy učení:

- *učení s učitelem* – vždy existuje nějaké vnější kritérium určující, který výstup je správný, a v síti se nastavují váhy pomocí zpětné vazby podle toho, jak blízko je výstup kritériu. Vypočítává se rozdíl mezi žádaným a skutečným výstupem. Váhy se nastavují podle nějakého algoritmu, který zabezpečuje snižování chyby mezi skutečným a žádaným výstupem. Metodika snižování rozdílu je popsána učícím algoritmem. Algoritmus může být jednokrokový, ale velmi často je iterační.

- *učení bez učitele* - nemá žádné vnější kritérium správnosti. Algoritmus učení je navržen tak, že hledá ve vstupních datech určité vzorky se společnými vlastnostmi. Do tohoto učení tedy není zapojen žádný vnější arbitr a celé učení je založeno pouze na informacích, které samotná síť během celého procesu učení získala. Říkáme tomu také samo-organizace. Příkladem sítě s učením bez učitele je Kohonenova neuronová síť.

2.4.2 Vybavování

Aktivní (vybavovací) fáze následuje za fází adaptivní. Z hodnot jednotlivých vstupů x_i a vah w_i se podle vzorce pro výpočet výstupu perceptronu (viz vzorec 2.1) vypočte odpovídající výstup neuronu. Pokud vybavování probíhá v síti, mění se jednotlivé výstupy neuronů tak dlouho, dokud nenastane rovnovážný stav.

Časově se mohou obě fáze překrývat, většinou se ale implementují odděleně. Frank Rosenblatt roku 1959 definoval (popsal) algoritmus pro nastavení perceptronů (též znám jako Perceptron convergence teorem) takto:

„*If there is a set of weights that correctly classify the (linearly seperable) training patterns, then the learning algorithm will find one such weight set,* w^* *in a finite number of iterations* „

Rosenblatt F (1962) *Principles of Neurodynamics*. Spartan, New York

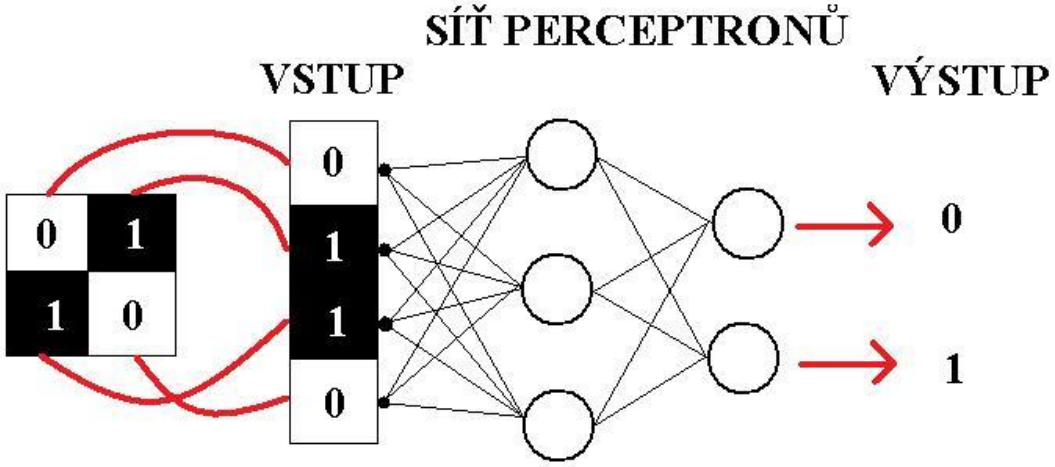
Volně přeloženo: Máme-li v n-rozměrném prostoru lineárně separabilní třídy objektů, pak lze v konečném počtu kroků učení (iterací optimalizačního algoritmu) nalézt vektor vah w perceptronu, který oddělí jednotlivé třídy bez ohledu na počáteční hodnoty těchto vah.

2.5 Metoda Back-Propagation

2.5.1 Uvedení

V této kapitole je popsána metoda s dopředným šířením informace a zpětným šířením chyby (feed-forward back-propagation). Je nejznámější a nejpoužívanější metoda ve vícevrstvých perceptronových sítích se snadnou implementací. Jde o metodu s učitelem, jejím hlavním cílem je minimalizovat chybu perceptronů skryté vrstvy gradientní metodou, respektive vyhledáním lokálního minima. Back-propagation dodržuje základní model třívrstvé perceptronové sítě.

Back-propagation se učí pomocí vzorů. Síti zadáme vzor, který požadujeme, aby byla schopna rozpoznat, a necháme algoritmus probíhat (měnit váhy, dle zjištěné chyby). Na výstup požadujeme rozpoznání právě toho vzoru (nyní může být i zašuměný. Sítě tohoto typu jsou velmi vhodné pro rozpoznávání sad znaků a pro mapovací úkoly.



Obr. 2.5 Ukázka možného vstupu a požadovaného výstupu

Pokud na vstup vložíme vzor tak jak je na Obr. 2.5, požadujeme na výstupu reprezentaci 0 a 1. O vzoru a jeho konkrétním výstupu říkáme, že je to *trénovací pár*. Jakmile je jednou síť naučena, poskytuje požadované výstupy pro jakýkoliv vstupní vzor.

2.5.2 Algoritmus

Celá síť je nejprve inicializována tak, že se všem vahám nastaví náhodné malé číslo, např. z intervalu <0,1>. Na vstup je přiveden vzor a vypočítán výstup celé perceptronové sítě (*feed-forward*) podle vzorce (2.1). Protože jsou hodnoty vah nastaveny náhodně, výsledek se po prvním průchodu naprosto liší od požadovaného. Proto vypočítáme chybu jako rozdíl požadované výstupní hodnoty (*Target*) a aktuálně získané výstupní hodnoty (*Output*) [9]:

$$Error = Output * (1 - Output) * (Target - Output) \tag{2.4}$$

Pro tuto konkrétní chybu byla použita aktivační funkce Sigmoida. Pokud použijeme jako aktivační funkci základní skokovou, výstupní chyba se vypočítá jako *(Target - Output)*. Tato varianta je popsána na začátku další kapitoly [**3.1.3**].

Přepočítáme pomocí této chyby všechny váhy (*back –propagation* chyby). W_{new} představuje novou, přepočítanou hodnotu váhy, $W_{current}$ aktuální hodnotu váhy[9]:

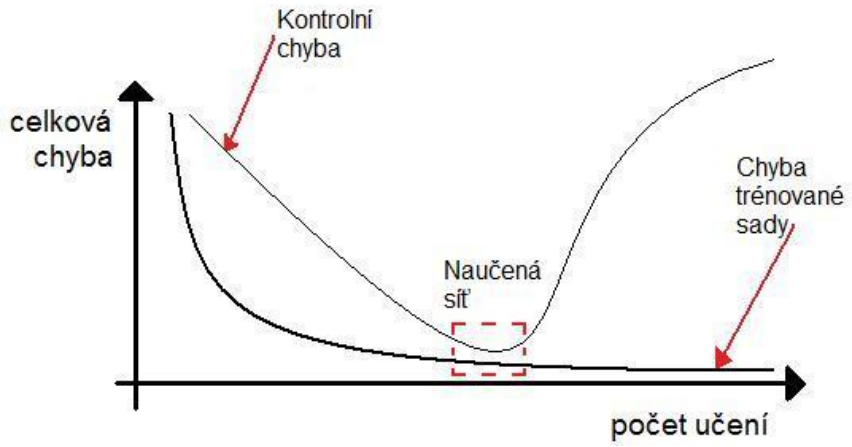
$$W_{new} = W_{current} + (Error * Input) \tag{2.5}$$

Výstup by se měl v dalším průchodu přiblížit požadovanému a tedy i chyba zmenšit. Celý postup opakujeme, dokud nedosáhneme požadovaného výsledku pro rozpoznávání. Z postupu je vidět, že cílem metody je změnit hodnoty vah tak, aby jako celek konvergovaly k minimu nějaké funkce.

Back-propagation pro skokovou aktivační funkci.
Algoritmus má jinou topologii, zcela chybí skrytá vrstva, a uplatňuje se *Hebbovo pravidlo*. To říká, že jestliže dva spolu propojené neurony jsou ve stejném okamžiku aktivní, pak jsou jejich vazby (představované synaptickými váhami) zesíleny. Pokud jsou oba neurony neaktivní, dochází k zeslabení jejich vzájemné vazby. V případě aktivity pouze jednoho neuronu, nejsou synapse modifikovány. Každý uvažovaný neuron má pouze dva stavy – aktivní a neaktivní.

2.5.3 Problémy metody Back-propagation

Velká přesnost (přeučení) – Metoda není schopna po dokončení učení rozpoznat zašuměná data. Problém nastává, pokud metodu učíme příliš dlouho. Hodnoty vah jsou zcela adaptovány na získání minima funkce a už nezohledňují vstupní sadu vzorů.



Obr. 2.6 Přesnost učení

Řešení spočívá v zavedení Kontrolní chyby. Ta po každém dokončeném cyklu učení přepočítá chybu v závislosti na vstupních vzorech. Algoritmus nepoužívá při učení zašuměná vstupní data, po zavedení kontrolní chyby je bude schopen ve vybavovací fázi rozpoznat.

Nalezení minima – Metoda se snaží měnit hodnoty vah tak, aby chyba klesala. Nastává ovšem problém, v případě, že chyba začne růst – došlo k nalezení lokálního minima funkce. Zde hrozí, že se metoda zasekne a chyba se nebude dále zmenšovat. Algoritmus chce ve vyhledávání pokračovat, chce dosáhnout globálního minima. Toho ale nedosáhne, protože není schopen procházet funkci směrem vzhůru. Existuje několik způsobů jak odstranit tento problém[9]:

- Všem vahám nastavíme nové náhodné hodnoty, metoda se bude muset učit znova a bude mít jinou funkci, ve které bude hledat minima

- Upravíme vzorce pro nastavení nových hodnot vah. Hodnoty nebudou závislé jenom na aktuální chybě, ale také na hodnotách z předešlého průchodu.

$$W_{new} = W_{current} + (Error * Input) + Momentum \tag{2.6}$$

Kde Momentum *M* vyjádříme následovně:

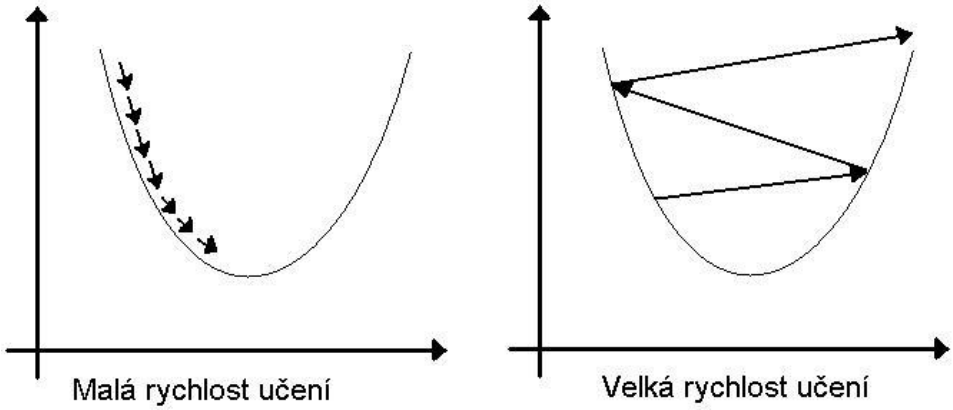
$$M * (W_{current} - W_{old}) \tag{2.7}$$

Momentum (setrvačnost) definované v intervalu <0,1> určuje, jak dlouho bude váhový vektor pokračovat ve stejném směru, než překročí hodnotu gradientního horizontu (než začne počítat gradient pro jiné minimum). Nevýhodou back-propagation s momentem je fakt, že pokud máme funkci, kde problém s lokálním minimem nenastává, pak je metoda bez momenta mnohem rychlejší.

Rychlost učení – Jeden z nejdůležitějších aspektů pro nalezení globálního minima. Značíme η a rovnice pro výpočet nových hodnot vah vypadá následovně[9]:

$$W_{new} = W_{current} + \eta(Error * Input) \tag{2.8}$$

Algoritmus s příliš malé hodnoty rychlosti učení probíhá velmi pomalu (konverguje velmi pomalu), naopak pro velké hodnoty rychle, ale hrozí riziko oscilace mezi relativně nezajímavými lokálními minimy, případně divergence u globálního minima.



Obr. 2.7 Rychlost učení

Obě tyto možnosti mohou být vyhledány pomocí experimentování a testování vzorků po pevně daném počtu průchodů. Běžně používané hodnoty rychlosti učení leží v intervalu <0,1>. Ideální by bylo, použít největší hodnotu rychlosti učení, při zachování schopnosti konvergovat k minimálnímu řešení.

2.5.4 Učení větších datových sad

V předchozích kapitolách jsme si popsali, jak úspěšně naučit metodu pro jeden vzor. Nyní popíšeme postup pro naučení sady vzorů – například čísel, písmen.

Na vstup vložíme první vzorek ze sady a provedeme veškeré výpočty a nastavení vah. Poté aplikujeme druhý, třetí až dokud nepoužijeme všechny vzory z celé sady. Celý proces opakujeme pro požadovaný počet průchodů – *epoch* (dále jen epocha). Je důležité, nejprve naučit celou sadu a teprve potom pokračovat další epochou. Kdybychom síť zcela naučili první vzor, a chtěli pokračovat na další, veškeré učení by bylo zbytečné. Hodnoty vah by byly po prvním průchodu pro druhý vzor přepsány a síť by nebyla schopna první vzorek rozpoznat. Proto je lepší sehnat si vzorovou sadu se stejným počtem vzorků pro každý symbol, v případě chybějícího vzoru v nějaké sadě pak opakovat vzorek reprezentující stejný symbol z předchozí sady. Algoritmus je nutné učit jednotlivé sady postupně, i když vzorky z různých sad reprezentující stejný symbol. V takovém případě se váhy přizpůsobují nejen jednotlivým vzorům, reprezentující různé symboly, ale pokouší se i najít určitou střední hodnotu pro vzory, reprezentující stejný symbol, z jiných vrstev.

U větších datových sad musíme algoritmu poskytnout dostatek epoch, aby byl schopen rozpoznat všechny vzory. Můžeme vypočítat celkovou chybu jako součet všech chyb neuronů a algoritmus ukončit po dosažení určité hodnoty, nebo nastavit přesnost, pro kterou se budou váhy měnit. Možností na ukončení je mnoho.

Jakmile je síť naučena, je schopna rozpoznat nejen přesné vzory, dle kterých se učila, ale i zašuměné a s chybami. Úspěšnost rozpoznání ještě můžeme vylepšit přidáním zašuměného vzoru do učící sady, případně náhodným proházením učících vzorů.

3 Návrh

Celá práce má za cíl popsat využití neuronových sítí ke zpracování obrazu, konkrétně pak k rozpoznání číslic. Požadujeme vytvoření takového systému, který by obsáhl problematiku jako celek. Kapitola popisuje návrhy na vytvoření systému schopného tento úkol řešit, je rozdělena do těchto podkapitol:

- Návrh systému pro učení
- Návrh systému pro rozpoznávání

K řešení zmíněného problému plně vyhovuje metoda Back-propagation. Návrhy zde popsané uvažují tuto metodu.

3.1 Návrh systému pro učení

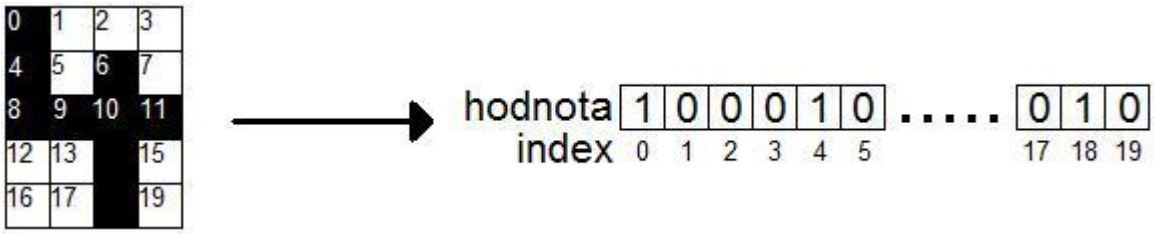
Základní jednotkou systému pro učení je bezesporu perceptron propojený vstupy a výstupy do perceptronové sítě. Z předchozích kapitol je zřejmé, s jakými typy informací pracuje. Datová struktura perceptronu tedy musí obsahovat informace o vstupech, vahách a výstupu. Struktura reprezentující perceptron může vypadat následovně:

```
struktura Perceptron{
double[] vstup;
double[] vaha;
double vystup;
};
```

Perceptrony můžeme nyní vložit do pole typu Perceptron[] a přistupovat k jednotlivým perceptronům pomocí indexů. Back-propagation funguje na modelu trojvrstvé sítě, definujeme tedy troje pole. Pole vstupů typu Perceptron[] a výstupů typu Perceptron[] a pole dvojrozměrné pro skrytou vrstvu Perceptron [][] (může být i jednorozměrné pokud skrytá vrstva obsahuje jednu vrstvu). Následují rozdíly a důvody použití polí v jednotlivých vrstvách.

3.1.1 Vstupní vrstva

Systém má rozpoznávat číslice, které jsou ve formě obrázků. Jakýkoliv obrázek si lze představit jako posloupnost pixelů s určitou barvou (v tomto případě s černou nebo s bílou, problematice barevných obrázků se věnuji později). Pro účely rozpoznávání, vyplníme vstupní pole vstup[] perceptronu hodnotami posloupnosti barev obrázku tak, že černá barva bude reprezentována číslem 1 a bílá číslem 0. Dostaneme tedy pole vstupu s hodnotami 1a 0 (případně 1.0 a 0.0, dle typu použitého pole).



Obr. 3.1 Reprezentace vstupní vrstvy

Nezáleží na pořadí, v jakém budeme do pole hodnoty zadávat. Je ale důležité, aby učení a rozpoznávání fungovalo podle jednoho systému.

3.1.2 Skrytá vrstva

Dvojrozměrné pole typu perceptron nabízí jednoduchou obsluhu a snadné ověřování správnosti. Problém, který může u skryté vrstvy nastat, je volba správného počtu perceptronů a vrstev. Neexistuje žádná metoda schopná najít optimální počet, je však dobré, držet se obecně známých doporučení. Počet perceptronů ve všech vrstvách, by neměl být menší než počet informací jdoucí jako vstup vstupní vrstvě a také, by neměl být více než dvojnásobný. Malý počet perceptronů, tedy i vah, vede k nedostatku váhových potenciálů a neschopnosti sítě si zapamatovat všechny vzory. Příliš mnoho vede zase k velké přesnosti, absolutní neúčinnosti vybavování a neschopnosti vůbec. Správný počet perceptronů je potřeba určit prakticky, provést několik cvičných učení a vyvodit výsledky.

3.1.3 Výstupní vrstva

Poslední vrstva určuje, kolik výstupů očekáváme a co je cílem adaptace pro konkrétní vzor ze sady. Perceptronová síť pro rozpoznávání čísel bude obsahovat 10 perceptronů na výstupní vrstvě (každý perceptron reprezentující jednu číslici), pro rozpoznávání písmen abecedy 26 perceptronů (uvažujme písmena anglické abecedy). Počet požadovaných výstupů tedy určuje počet perceptronů ve výstupní vrstvě.

Dále každému konkrétnímu vzoru musíme po jednom skončeném průchodu sítí určit chybu získaného výstupu (*out*) od požadovaného (*target*). Je dostačující, pokud výstup vedoucí k požadovanému správnému perceptronu ohodnotíme jako 1 a ke špatnému 0. Tato metoda zabezpečuje správné zpětné šíření chyby a správné přepočítávání vah. Existuje několik přístupů k vypočtení chyby. Základní metoda back-propagation používá sumu všech chyb vektorů výstupů výstupní vrstvy ke zjištění chyby u jednotlivých výstupních neuronů:

$$chyba = \frac{1}{2}\sum(target - out)^2 \tag{3.1}$$

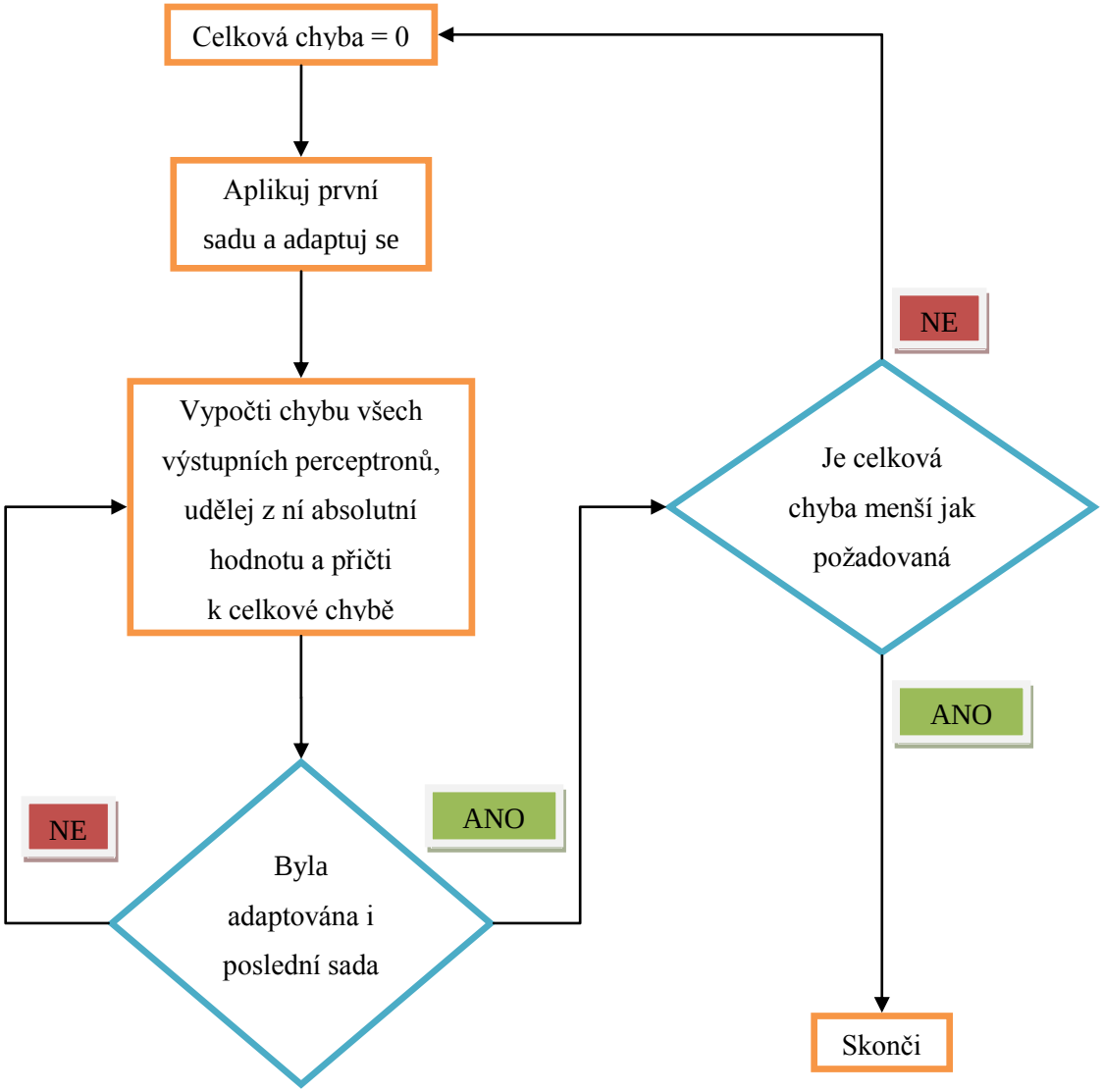
Back-propagation požívající sigmoidu jako aktivační funkci zase uplatní následující vzorec[9]:

$$chyba = out(1 - out)(target - out) \tag{3.2}$$

kde *out (1 - out)* je první derivací funkce sigmoidy.

3.1.4 Algoritmus

Obecný algoritmus pro adaptaci metody Back-propagarion lze popsat pomocí následujícího diagramu:



Obr. 3.2 Diagram pro Back-propagation

3.2 Návrh systému pro rozpoznávání

Rozpoznávání není příliš vhodné posuzovat jen z pohledu perceptronových sítí. Mohou nastat situace, kdy je lepší obraz různými metodami upravit před samotným rozpoznáváním. Z praktického života víme, že takové situace nastávají neustále (skenování psaného textu, identifikace poznávací značky rychle jedoucího auta). Na problematiku jsem se proto rozhodl nahlížet z vyšší perspektivy, z pohledu oboru zpracování obrazu.

Back-propagation je schopen rozpoznávat i barevné obrázky, jeho implementace však není tak triviální (místo binárních hodnot pixelu se například vkládají přímo vektory, je potřeba si uchovávat hodnoty sousedních pixelů atd.) Existují metody, s jejichž pomocí lze dosáhnout velmi kvalitních výsledků. Nejsou tak efektivní jako backpropagation pro barevný vstup, ale pro tyto potřeby plně dostačují. Nejzákladnější je použití prahování (tresholding).

3.2.1 Prahování

Každý bod obrazu je reprezentován hodnotou, která je kombinací tří barev (uvažujme standardní RGB model). Intenzita (jas) bodu lze vypočítat pomocí vzorce X. Jde vlastně o převod do stupňů šedi pomocí poměrného zastoupení jednotlivých složek barev [11].

$$I = 0.299R + 0.587B + 0.114G \tag{3.3}$$

Prahování je metoda, která dle intenzity rozhodne, zda pixel vynulujeme, pokud nepřesahuje stanovenou hodnotu prahu, nebo nastavíme na hodnotu černé. Prahovaný obraz má právě jednu hodnotu prahu. Ta může být nastavena pevně (např. na hodnotu středu pro převod na černobílý orázek) nebo dle histogramu – globální prahování.

Problém globálního prahování spočívá v tom, že není schopno reagovat na lokální změny v jasové funkci. Získaná hodnota prahu pak nemusí být optimální ve všech částech obrazu a nastávají komplikace, např. tmavé skvrny po krajích obrazu podobně těm při skenování zvlněného papíru.

Možné řešení je adaptivní prahování, které se snaží nalézt hodnotu prahu na malém okolí pixelu.

3.2.2 Zjištění úhlu a pootočení

Návrh se bude skládat ze dvou částí, nejprve musíme zjistit, o jaký úhel se bude obrázek otáčet, teprve pak můžeme samotné otočení provést.

K vyhledání úhlu použijeme Houghovu metodu (transformaci). Vyhledává parametrický popis objektů v obraze. Při implementaci je třeba znát analytický popis tvaru hledaného objektu (v našem případě přímky). Metoda je používána především pro segmentaci objektů, jejichž hranici lze popsat právě zmíněnými křivkami. Hlavní výhodou je schopnost rozpoznat i objekt s nepravidelnostmi, zašumělý objekt a objekt s chybějícími částmi. Metoda je výpočetně velmi náročná. Je důležité si uvědomit, že bude fungovat jen pro nějaký soubor znaků, např. text. Pro jeden samotný znak můžeme obrázek pootočit o několik stupňů a zkusit rozpoznat, pak opět pootočit a tak

pokračovat dokud neopíšeme kružnici nebo dokud je chyba rozpoznání tak malá, že můžeme považovat rozpoznávání za úspěšné.

Pootočení [12] se provádí pro každý pixel samostatně. Využijeme k tomu počátku souřadnicového systému, kdy úhel otočení určujeme podle něj. Nejprve pixel *P* převedeme do homogenního tvaru:

$$P = [x, y] \longrightarrow P = [x, y, w] \tag{3.4}$$

Kde *w* je váha, kterou vynásobíme souřadnice *x* a *y*. A transformujeme pomocí matice *M*:

$$M = \begin{pmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.5}$$

Nové souřadnice pixelu tedy budou:

$$P' = M * P \tag{3.6}$$

Získali jsme novou pozici pixelu. Pro naše účely je však lepší využít rotaci podle zvoleného středu (středu stránky). Použijeme počátku jako bodu, kolem kterého se bude rotace provádět. Pixel tedy pomocí transformační matice přesuneme k počátku, otočíme a přesuneme zpět na původní místo.

$$T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \delta x & \delta y & 1 \end{pmatrix} \tag{3.7}$$

I když nebude pootočení úplně přesné, perceptronová síť by měla znak poznat. V případě textu je potřeba ještě provést segmentaci, tzn. oddělení jednotlivých znaků.

3.2.3 Segmentace

Pro určení pozice znaku pro rozpoznávání použijeme opět převod do stupňů šedi. Konkrétněji použijeme vertikální a horizontální histogram stupňů šedi v celém obraze. Vertikální nám oddělí řádky a horizontální znaky a mezery. Určíme si prahovou hodnotu, podle které budeme považovat data z histogramů za znak a naopak za mezeru, řádek nebo šum. Můžeme tedy jednotlivé znaky vysekávat a posílat na analýzu perceptronové síti, známe totiž jejich souřadnice. Problém může nastat u spojených nebo jinak deformovaných znaků. Tato problematika je však zcela nad rámec této práce.

Obr. 3.3 Vertikální a horizontální histogram

4 Implementace a výsledky

Součástí bakalářské práce je program, schopný rozpoznat čísla pomocí metody back-propagation. V této kapitole popíši implementovaný systém s názornou ukázkou výsledků. Program jsem rozdělil do tří hlavních částí:

- Učení dle vzoru
- Rozpoznání jedné číslice
- Rozpoznání sady číslic s určením pozice na papíře

Pro testování jsem použil několik sad číslic. Základem je jednoduchá sada deseti číslic vytvořená v malování, pro ověření správné funkčnosti algoritmu. Tuto sadu jsem různě upravoval a přidával šum, posunoval pozice apod. Další sady číslic jsou převážně převzaty z různých typů písem (fontů), dále pospojovány pro vytvoření souboru sad. Je možné vkládat i barevné číslice, program je ale prahuje dle střední hodnoty 127 (rozsah 0-255). Samozřejmostí je vkládání různých typů souborů obrázků: bmp, jpeg (jpg), png, gif, tiff. Pokud nebude řečeno jinak, jako standardní budu používat formát bmp. U ostatních jsem díky různým metodám komprimací, úpravy barev apod. dosáhl nežádoucích efektů. Jako nejméně vhodný (zejména pro adaptaci) se pak ukázal formát jpeg, právě kvůli zmíněné metodě prahování. Ve fázi rozpoznávání jsem proto pro jistotu implementoval nastavitelnou hodnotu prahu. V následujících kapitolách je pro názornost uvedeno jaké sady, případně formátu, bylo použito.

4.1 Učení dle vzoru

Program je implementován tak, aby poskytl uživateli maximální možnou volbu přednastavení. Taková flexibilita je nezbytná při samotném testování a určování nejlepších výsledků. Při dosažení vhodných výsledků je možné neuronovou síť uložit pro pozdější použití. Je nutné opět podotknout, že každé nové učení nastavuje nové hodnoty vah neuronům. Pro co možná největší objektivnost bylo nutné některé učení provádět opakovaně tak, aby se celkové výsledné chyby co nejvíce blížily (funkce hodnot vah měly podobné minimum).

Jako základní vzorovou sadu jsem použil vlastní obrázky 5x7 pixelů reprezentující 10 číslic. Záměrně jsem se snažil použít takovou sadu, jejíž číslice by se navzájem podobaly. Obrázky reprezentující číslice 5 a 6 se liší pouze ve dvou pixelech.

Obr. 4.1 Číslice 5 a 6

Nastavení sítě

Počty neuronů: 7 ve vstupní vrstvě
7 v jedné skryté vrstvě
10 ve výstupní vrstvě

Aktivační funkce: sigmoida
Vstup: 10 obrázků s rozměry 5*7 pixelů reprezentující 10 číslic

Výstup porovnávám se vzorovými obrázky číslic 5 a 6.

Výsledky testů pro různé hodnoty počtu epoch, rychlostí učení, zvolené přesnosti. Vždy je provedena sada dvaceti testů, je vybrán nejlepší a nejhorší výsledek (největší a nejmenší celková chyba), jde vlastně o součet chyb ve všech vrstvách sítě. Porovnávám úspěšnost (hodnotu na výstupu výstupní vrstvy) na dvou nejvíce si podobných číslicích, konkrétně 5 a 6.

4.1.1 Testy počtu epoch

Počet epoch	Nejmenší celk. chyba	Rozpoznal 5	Úspěšnost pro 5	Rozpoznal 6	Úspěšnost pro 6
4000	1.35638	NE - 9	0.08876	NE - 9	0.09124
40 000	0.62564	ANO	0.66674	ANO	0.81406
400 000	0.02919	ANO	0.949835	ANO	0.968715

Tabulka 4.1 Test počtu epoch s nejmenší celkovou chybou pro 20 testovaných sítí

Počet epoch	Největší celk. chyba	Rozpoznal 5	Úspěšnost pro 5	Rozpoznal 6	Úspěšnost pro 6
4000	1.79476	NE - 4	0.10804	Ne - 4	0.11729
40 000	1.20086	ANO	0.59278	ANO	0.45729
400 000	0.03770	ANO	0.95167	ANO	0.947696

Tabulka 4.2 Test počtu epoch s největší celkovou chybou pro 20 testovaných sítí

4.1.2 Vyhodnocení testu počtu epoch

Výsledné testy dopadly dle předpokladu a potvrdily teorii. Pro malý počet epoch se váhy nebyly schopny adaptovat a číslice nebyly rozpoznány. Hodnoty úspěšnosti u ostatních číslic se pohybovaly na stejném číselném řádu nebo o řád níže. Tento počet epoch je nedostatečný pro rozpoznání.

Pro 40 000 epoch dopadly testy lépe. Číslice byly rozpoznány. Váhy se stačily dostatečně adaptovat. Je zajímavé, že při tomto počtu epoch hraje roli největší / nejmenší celková chyba naučené

sítě. Z tabulky můžeme vidět znatelný rozdíl v úspěšnosti. I přes tento rozdíl lze považovat síť za schopnou rozpoznat číslice, chyby pro ostatní číslice se pohybovaly ve stejném desetinném řádu, maximálně o dva řády níž, proto může dojít k chybě rozpoznání u zašuměných dat.
Epochy s největším testovaným počtem průchodů dopadly nejlépe. To zajisté potvrzuje i úspěšnost. Ta se u ostatních číslic pohybovala o jeden až šest desetinných řádů níže. U největších rozdílů jsou tedy číslice pro takovouto síť nezaměnitelné. Problémem této sítě byla velmi dlouhá doba učení. Pokud bychom použili vzorový obrázek s desetinásobnými rozměry (samozřejmě i jinou neuronovou síť, viz podkapitola **3.1.2**), doba trvání adaptace by neúměrně stoupla na několik hodin.

Síť byla naučena s menším počtem neuronů (24), než je počet vstupů (35). Rozhodl jsem se proto, provést ještě několik testů, zohledňující tuto skutečnost. Vstupní vrstva obsahovala 12 neuronů, jedna skrytá vrstva 12 a výstupní 10. Pro počet epoch 4000 dopadly testy v porovnání s původními o něco lépe, nejmenší nalezená chyba dosahovala hodnoty 0,98, rozdíl oproti hodnotám z tabulky 4.1 činí nezanedbatelných 0,4. Přesto síť nebyla dostatečně adaptována pro rozpoznávání. Nepodařilo se úspěšně rozpoznat žádnou číslici. Zkoumat podrobně závislosti počtu neuronů a topologii budu později.

Další testy jsou provedeny s hodnotou počtu epoch 80 000. Tento počet nepochybně bude stačit, vzhledem k získaným výsledkům (uvažuji podobný počet neuronů).

4.1.3 Testy rychlosti učení

Stejná konfigurace sítě jako předchozí test, počet epoch 80 000.

Rychlost učení	Nejmenší cel. chyba	Rozpoznal 5	Úspěšnost pro 5	Rozpoznal 6	Úspěšnost pro 6
0.5	0.18776	ANO	0.92109	ANO	0.90257
1	0.28514	ANO	0.82546	ANO	0.87166
2	0.17810	ANO	0.922089	ANO	0.89164

Tabulka 4.3 Test rychlosti učení s nejmenší celkovou chybou pro 20 testovaných sítí

Rychlost učení	Největší cel. chyba	Rozpoznal 5	Úspěšnost pro 5	Rozpoznal 6	Úspěšnost pro 6
0.5	0.44904	ANO	0.81419	ANO	0.89164
1	0.46522	ANO	0.86241	ANO	0.90030
2	0.40748	ANO	0.72196	ANO	0.81599

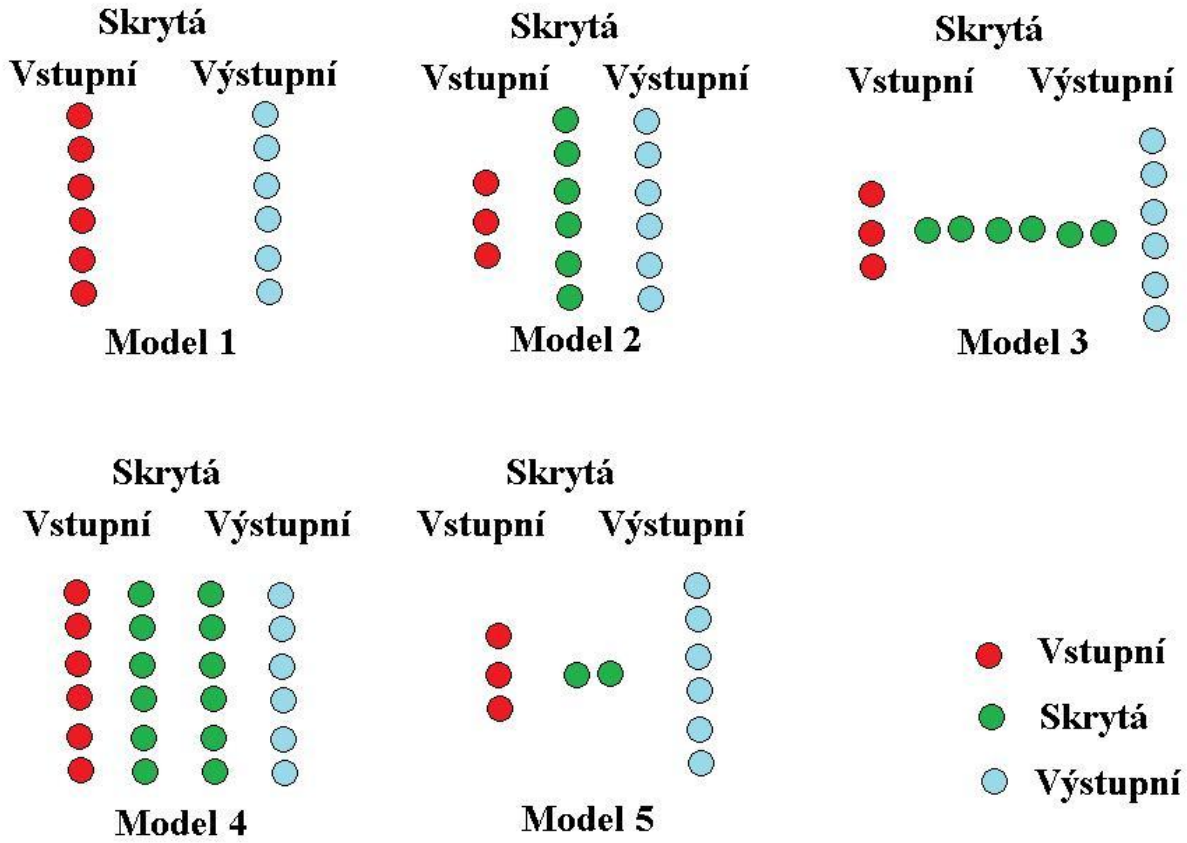
Tabulka 4.4 Test rychlosti učení s největší celkovou chybou pro 20 testovaných sítí

4.1.4 Vyhodnocení testu rychlosti učení

Rychlost učení nemá vliv na takovéto konkrétní nastavení sítě, pro všechny hodnoty dopadl test téměř shodně. V průběhu testu došlo v jednom případě k absolutní odchylce, kdy se největší celková chyba pohybovala kolem hodnoty 1,7. Tuto hodnotu jsem anuloval. Dále bylo zjištěno, že pokud se hodnota celkové chyby pohybuje kolem 0.2, pak je úspěšnost pro rozpoznání čísla velmi vysoká. Mám tedy jistotu, že hodnota počtu epoch byla vybrána vhodně. Neuronové sítě z tabulky jsou uloženy na přiloženém médiu.

4.1.5 Testy závislostí počtu vrstev a počtu neuronů

V této kapitole zkusím ověřit teorii z kapitoly návrhy. Otestuji funkčnost základního vertikálního a horizontálního modelu skryté vrstvy, jejich kombinace, model bez skryté vrstvy a schopnost sítě poradit si s nedostatkem a přebytkem počtu neuronů. Stále uvažuji počet 80 000 epoch, budu testovat 20 sítí na jeden model, dokud nedosáhnu počtu epoch nebo nedosáhnu minimální celkové chyby 0.25. Celkový počet neuronů v testech je 30. Nejúspěšnější model budu používat v dalších testech rozpoznávání.



Obr. 4.2 Testování topologie a počtu

Model 1
Neuronovou síť bez skryté vrstvy. Vstupní vrstva 25 neuronů, výstupní 10.

	Celková chyba	Počet epoch	Rozpoznal 5	Úspěšnost pro 5	Rozpoznal 6	Úspěšnost pro 6
Nejlepší výsledek	0.34646	80 000	ANO	0.88934	ANO	0.26432
Nejhorší výsledek	0.62357	80 000	ANO	0.88082	NE - 8	0.42968

Tabulka 4.5 Výsledky pro Model 1

Tetování skončilo po dosažení počtu epoch. Z výsledků byla zřejmá podobnost obou číslic. Hodnoty úspěšnosti se pohybovaly na stejném desetinném řádu, ostatní o jeden až tři řády níže.

Model 2
Test horizontálního modelu. Vstupní vrstva 5 neuronů, jedna skrytá vrstva 20 neuronů, výstupní 10.

	Celková chyba	Počet epoch	Rozpoznal 5	Úspěšnost pro 5	Rozpoznal 6	Úspěšnost pro 6
Nejlepší výsledek	0.25	35 945	ANO	0.89597	ANO	0.90480
Nejhorší výsledek	0.25	71 501	ANO	0.89082	ANO	0.80580

Tabulka 4.6 Výsledky pro Model 2

Při testu bylo dosaženo velmi uspokojivých výsledků. Přestože při adaptaci docházelo k častějším divergencím (4 z 20), síť se při konvergenci perfektně adaptovala. Rozdíly mezi identifikovanými čísly 5 a 6 tvořily téměř jeden celý desetinný řád. Tento model je vhodným kandidátem pro rozpoznávání.

Model 3
Test vertikálního modelu. Vstupní vrstva 5 neuronů, 20 skrytých vrstev po jednom neuronu, výstupní 10 neuronů.

	Celková chyba	Počet epoch	Rozpoznal 5	Úspěšnost pro 5	Rozpoznal 6	Úspěšnost pro 6
Nejlepší výsledek	1.59332	80 000	NE - 9	0.09911	NE - 9	0.09936
Nejhorší výsledek	1.59619	80 000	NE - 9	0.09921	NE - 9	0.09943

Tabulka 4.7 Výsledky pro Model 3

Testování ukončeno po pěti náhodných sítích. Již během testu se celková chyba pro každou novou epochu měnila na šestém desetinném místě. Tato topologie je zcela nevhodná pro další použití.

Model 4
Test modelu s různým počtem skrytých vrstev. Pro názornost provedu dva testy.

Test 1: Vstupní vrstva 5 neuronů, 2 skryté vrstvy po deseti neuronech, výstupní vrstva 10 neuronů.

	Celková chyba	**Počet epoch**	**Rozpoznal 5**	**Úspěšnost pro 5**	**Rozpoznal 6**	**Úspěšnost pro 6**
Nejlepší výsledek	1.202505	80 000	NE - 9	0.083334	NE - 9	0.08721
Nejhorší výsledek	1.225808	80 000	NE - 9	0.084447	NE - 9	0.08812

Tabulka 4.8 Výsledky pro Model 4, Test 1

Test 2: Vstupní vrstva 5 neuronů, 4 skryté vrstvy po pěti neuronech, výstupní vrstva 10 neuronů.

	Celková chyba	**Počet epoch**	**Rozpoznal 5**	**Úspěšnost pro 5**	**Rozpoznal 6**	**Úspěšnost pro 6**
Nejlepší výsledek	1.43255	80 000	STORNO	STORNO	STORNO	STORNO
Nejhorší výsledek	1.44004	80 000	STORNO	STORNO	STORNO	STORNO

Tabulka 4.9 Výsledky pro Model 4, Test 2

Provedené testy nedopadly dobře, druhý test musel být po pěti pokusech ukončen. Po 5000 pokusech bylo zřejmé, že tato topologie konverguje k minimu velmi pomalu. U obou docházelo jen k malé změně celkové chyby. Oba případy se velmi podobají vertikálnímu modelu.

Z výsledků modelů jedna až čtyři tedy vyplývá, že sítím tohoto typu vyhovuje méně vrstev, zato s větším počtem neuronů. Pro další testy jsem se tedy rozhodl použít model 2.

Model 5
V posledním modelu testuji vliv neadekvátního počtu neuronů ve skryté vrstvě vzhledem ke vstupu. Je požadováno 35 neuronů. Test 1 zaměřím na nedostatek a test 2 na přebytek. Jako topologii jsem zvolil model 2.

Test 1: Vstupní vrstva 5 neuronů, jedna skrytá vrstva se dvěma neurony, výstupní vrstva 10 neuronů. Celkem 17 neuronů, polovina požadavku.

	Celková chyba	**Počet epoch**	**Rozpoznal 5**	**Úspěšnost pro 5**	**Rozpoznal 6**	**Úspěšnost pro 6**
Nejlepší výsledek	2.27651	80 000	STORNO	STORNO	STORNO	STORNO
Nejhorší výsledek	2.29951	80 000	STORNO	STORNO	STORNO	STORNO

Tabulka 4.10 Výsledky pro Model 5, Test 1

Test byl stornován, metoda konvergovala velmi krátce, značně divergovala s celkovou chybou přesahující 2. Je ale velmi zajímavé, že změna celkové chyby po jednotlivých epochách byla mnohem větší než v modelu 3.

Test 2: Vstupní vrstva 5 neuronů, jedna skrytá vrstva se 37 neurony, výstupní vrstva 10 neuronů. Celkem 52 neuronů, o polovinu více neuronů.

	Celková chyba	**Počet epoch**	**Rozpoznal 5**	**Úspěšnost pro 5**	**Rozpoznal 6**	**Úspěšnost pro 6**
Nejlepší výsledek	0.25	37731	ANO	0.83611	ANO	0.92889
Nejhorší výsledek	0.30343	80 000	ANO	0.88749	ANO	0.86450

Tabulka 4.11 Výsledky pro Model 5, Test 2

Metoda konvergovala a divergovala velmi prudce, a pokud dosáhla požadované celkové chyby resp. počtu epoch, rozpoznávala číslice s velkou přesností. Nepochybně hlavním faktorem ovlivňující toto chování je moment nastavení počátečních vah. Pokud jsou váhy nastaveny dostatečně náhodně tedy vhodně, můžeme i pro velký počet neuronů po krátkém časovém úseku získat požadované řešení. Velmi užitečné by pak bylo použít nadbytek neuronů pro větší rozměry vstupu. Prudkost a náhlost konvergence by zaručila řešení. Vše je nutné ověřit testováním, síť musí projít určitým počtem epoch s naučením.

4.1.6 Vyhodnocení testů

U většiny neúspěšných testů vidíme, že síť rozpoznává obraz jako číslici 9. Je to zapříčiněno nedokončenou adaptací. Algoritmus prochází výstupy a vypočítává výstupní chybu sekvenčně tak, že pokud nastal problém s adaptací, nejlepší šanci na úspěch má vždy 9 (např. Model 4, Test 1). S každou další epochou se výstupní chyba u všech číslic vyrovnává. Síť se dostává do fáze, kdy už je schopna číslice rozpoznat. Úspěšnost takové identifikace bývá v průměru kolem 0,6. Taková síť ještě není připravena dostatečně (Model 1). Jako dostatečně naučenou můžeme klasifikovat síť, jejíž celková chyba je menší jak 0.25 (Model 2). Taková síť by měla být schopna rozpoznávat i silně zašumělá data. Více v další kapitole.

Nejdůležitější poznatek z této kapitoly je však zjištění, že pokud obsahuje neuronová síť alespoň podobný počet neuronů, jaký je počet vstupních informací, pak nezáleží ani tak na počtu neuronů, jak na správné topologii. Test horizontálního modelu (Model 2) téměř vždy končil před hraničním počtem epoch s dostatečně malou celkovou chybou. Tento model budu používat i v dalších testech.

Všechny testy úspěšně proběhly i pro větší rozměry obrázků. Bylo potřeba provést několik zkušebních epoch pro získání představy fungování sítě pro větší počty neuronů.

Na přiloženém CD je možné najít přednastavené sítě pro testy jednotlivých modelů.

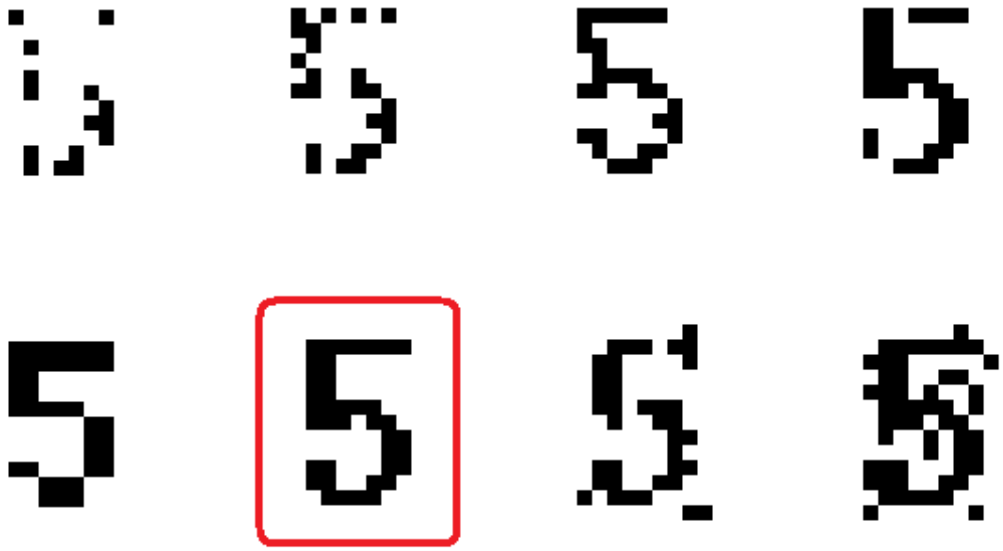
4.2 Rozpoznání jedné číslice

V této kapitole budu dále testovat neuronovou síť, zaměřím se na samotné rozpoznávání. Pokusím se testovat různě zašumělé datové vstupy, s tím související nežádoucí vlastnost sítě – přílišnou adaptaci (přeučení), nakloněný obraz apod.

Raději zvolil rozměrnější objekt. Vzorem je znaková sada MS PMincho s velikostí písma 11 bodů, obrázky v rozměru 9 x 13 pixelů. Neuronová síť třívrstvá, vstupní vrstva 20 neuronů, jedna skrytá vrstva 100 neuronů a výstupní vrstva 10 neuronů.

4.2.1 Šum v obraze

Testuji výskyt nežádoucích objektů v obraze.



Obr 4.3 Vzorová sada se šumem

Červeně je vyznačen původní obrázek.

Výstupní chyba	0.499437	0.983234	0.974127	0.991593
Rozpoznal	ANO	ANO	ANO	ANO
Výstupní chyba	0.011060	**0.992629**	0.948181	0.981092
Rozpoznal	ANO	**ANO**	ANO	ANO

Tabulka 4.12 Zašuměná data

Zjistil jsem, že metoda velmi úspěšně rozpoznává zašuměné datové sety. V pátém testu byla číslice úspěšně rozpoznána i s velmi malou výstupní chybou. V dalším testu budu zjišťovat schopnost rozpoznat posunuté číslice.

4.2.2 Posunutí v obraze

Vzor jsem posouval o jeden pixel doleva.

Posunutí vlevo	1 pixel	2 pixely	3 pixely
Výstupní chyba	0.982462	0.017535	0.022967
Rozpoznal	ANO	NE - 7	NE - 7

Tabulka 4.13 Posunutí v obraze

Testováním jsem dokázal, že metoda není vhodná k identifikaci posunutých objektů oproti vzorové sadě. Před rozpoznáváním je potřeba objekt vycentrovat nebo použít jiné metody zpracování obrazu.

Posunutí pro jeden pixel dopadlo úspěšně, protože šířka původního vzorového obrázku byla 2 pixely. Síť tedy rozpoznávala pětku se šumem.

4.2.3 Nakloněný obraz

Při naklánění do obrázku přibyly odstíny šedi, u rozpoznávání jsem proto požil metodu prahování k odstranění tohoto jevu. Testovaný obrázek je pootočen o 5 stupňů a každý o dalších 5. Rotuji po směru hodinových ručiček.

Obr. 4.4 Nakloněný obraz

Rotace ve stupních	+5°	+10°	+15°	+20°	+25°	+30°
Hodnota prahu	127	127	115	109	109	107
Výstupní chyba	0.207249	0.585939	0.013061	0.006332	0.009635	0.021520
Rozpoznal	ANO	ANO	NE - 6	NE - 6	NE - 6	NE - 4

Tabulka 4.14 Nakloněný obraz

Test potvrdil předpoklad, že nakloněná pětka je snadno zaměnitelná se šestkou. I s různými hodnotami prahu jsem dosahoval stejných výsledků, pouze pro rotaci +30°, síť identifikovala obrázek jako čtyřku pro prahové hodnoty 100 ± 20, pro ostatní opět jako šestku.

Problém se zaměnitelností může být způsoben i drobným posunutím. Je nutné přesně určit střed vzoru a teprve rotovat. Posunutí ve výsledku způsobí velmi malé výstupní hodnoty pro požadované číslo. Obecně jsem dosahoval malých výstupních hodnot pro všechny čísla.

4.2.4 Přílišná adaptace

Síť budu testovat pro celkový počet 1 000 000 epoch a pokusím se dokázat existenci tohoto problému.

Testováním i s takto velkým počtem epoch neprokázalo efekt přeučení. Pokusím se test opakovat s menší znakovou sadou a jinou neuronovou sítí. Použiji stejnou jako v Modelu 2 kapitoly **4.1.5.**
Počet epoch 10 000 000.

Ani výsledky tohoto testu nezjistili přítomnost přeučení. Výstupní hodnota vzoru pro pětku byla 0.997, se šumem 0.995. Obrázky se přitom lišily v 6 pixelech z 35 (téměř 20% chyba).

Problém přílišné adaptace se mi nepodařilo potvrdit.

4.2.5 Vyhodnocení testů rozpoznání jedné číslice v obraze

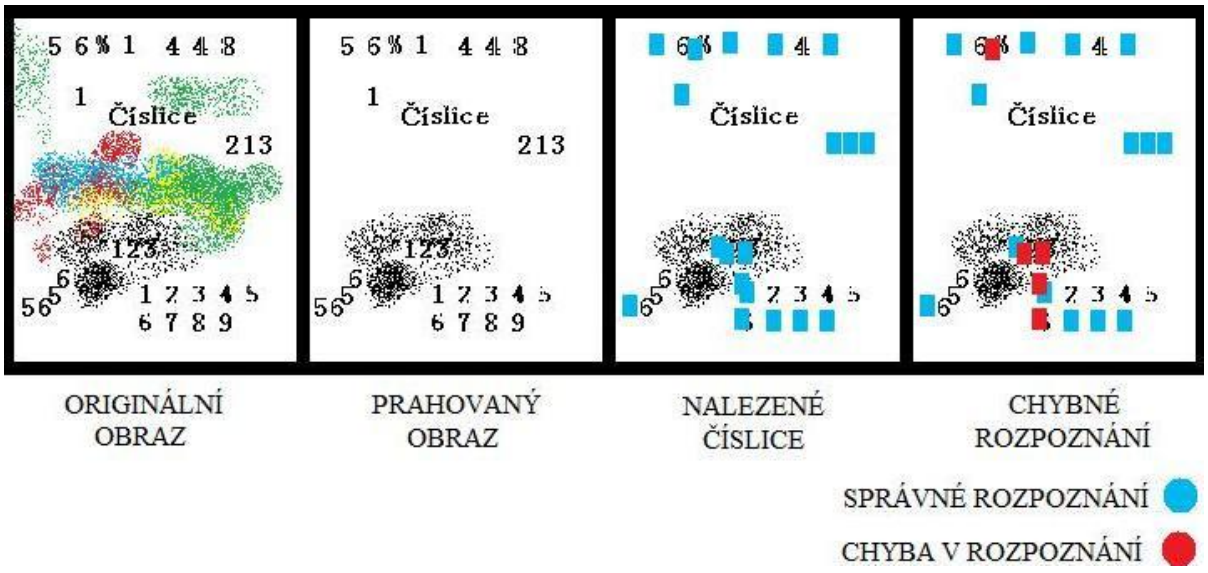
Algoritmus Back-propagation výborně rozpoznává obraz se šumem. I hodně znehodnocené číslice se podařilo identifikovat s velkou úspěšností. Problém nastal, pokud byl obraz nějakým způsobem vychýlen, ať už posunem, nebo rotací. Poměrně úspěšně byla síť schopna rozpoznat mírně pootočené číslice. Ostatní se nedařilo. Jde o důležité zjištění, které je potřeba si uvědomit, pokud budeme chtít rozpoznávat objekty ve větším obraze (viz kapitola 3.2 Návrh systému pro rozpoznávání). V další části se takové rozpoznání pokusím provést.

4.3 Rozpoznání sady číslic s určením pozice na papíře

V poslední části se pokusím otestovat schopnost programu rozpoznat čísla na papíře a v obraze. Dle zjištění nemožnosti identifikovat posunuté znaky jsem se rozhodl, že celý obrázek budu procházet po jednotlivých pixelech a podle úspěšnosti budu symboly vyhodnocovat. Neuvažuji zcela otočených symbolů, předpokládám správné vložení obrázku (uživatel vložil papír do skeneru podle návodu). Takové rozšíření je možné implementovat později, i tak je tato část jistým nadstandardem zadané bakalářské práce. Chtěl jsem ověřit schopnost adaptovat se na problémy z praxe.

4.3.1 Obecné rozpoznání

Vytvořil jsem obrázek se vzory dle předchozích kapitol. Použitá neuronová síť se shoduje s použitou v kapitole **4.2**.

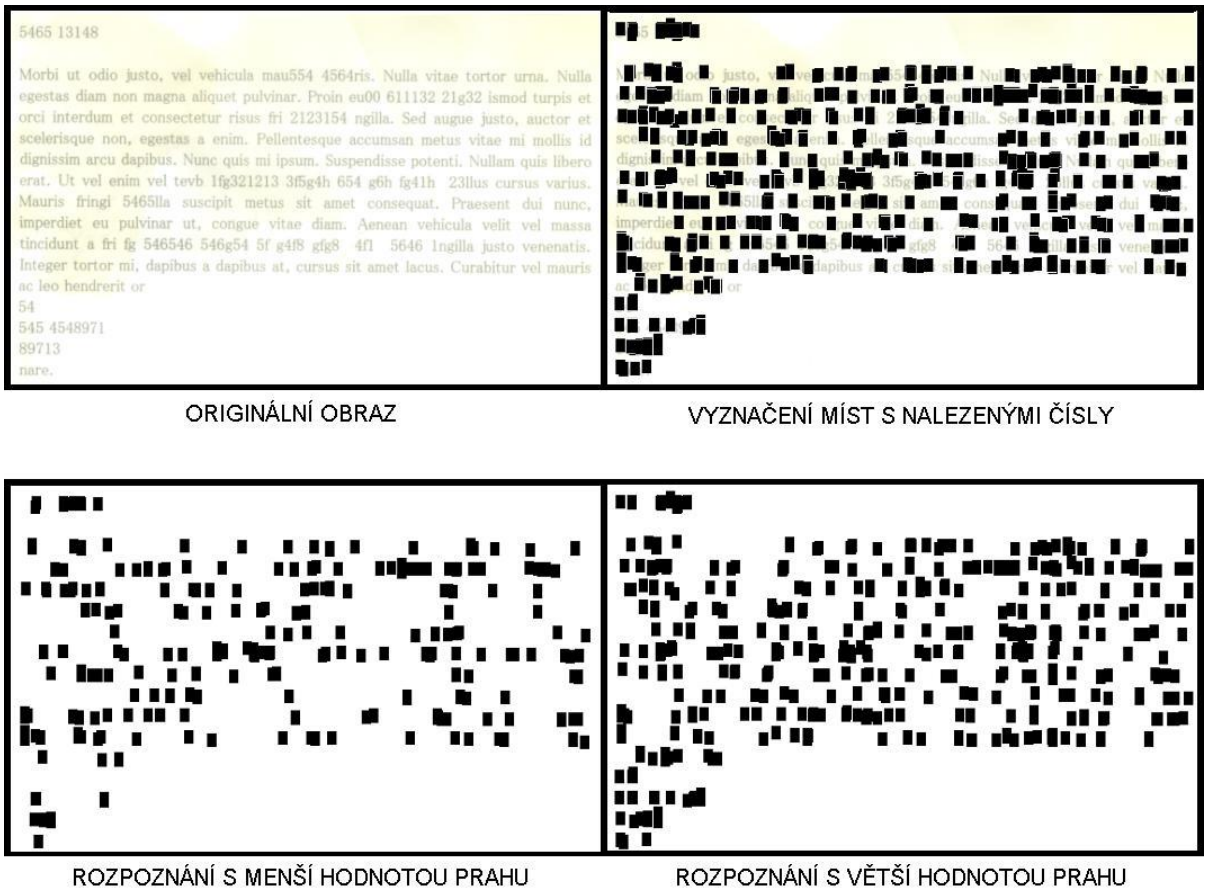


Obr 4.5 Obecné rozpoznání

V obrázku bylo nalezeno 20 objektů, z nichž 6 bylo rozpoznáno chybně a 14 správně, i s pozicí na papíře. Úspěšnost rozpoznání správného rozpoznání z nalezených objektů je tedy 70%. Tvrdit, že je metoda úspěšná, zatím nelze. Obrázek je vytvořen na počítači a nezohledňuje problémy v praxi. Další test provedu na naskenovaném dokumentu.

4.3.2 Rozpoznání naskenovaného dokumentu

Neuronová síť je naučena vzory písmen řady MS PMincho. Vybranému dokumentu jsem celý text převedl do uvedeného druhu písma, dokument vytisknul, polil nápojem, pomačkal a naskenoval. Následně byla aplikována neuronová síť.



Obr. 4.6 Rozpoznání naskenovaného dokumentu

Vyhodnocení

Snažil jsem se docílit věrohodnosti tím, že maximálně přiblížím testování běžnému kancelářskému prostředí. Naskenovaný obraz po rozpoznání nevykazoval velkou úspěšnost vyhledávání, bylo tedy nutné chvíli hledat vhodnou hodnotu při prahování. Každé nové rozpoznání pro jiné hodnoty prahu označily i jiné místa výskytu číslic. Některá místa výskytu se však opakovala a jak je vidět na obrázku 4.6, takové lokace skutečně číselné hodnoty obsahují (pravý horní obrázek je sloučením všech tří ostatních dohromady). Existovaly číslice vykazující velmi slušnou stabilitu i při změně vah.

Pro rozpoznávání číslic v obrázku takovéhoto typu je zcela nezbytné jej nejprve předpřipravit, např. použitím metod zvětšení ostrosti obrazu, adaptivním způsobem prahování apod.

4.3.3 Rozpoznání číslic v obraze

Pro identifikaci jsem zvolil fotku.

Obr. 4.7 Rozpoznání číslic v obraze

Podobně jako u rozpoznání textu, metoda spíše než nalezení konkrétních hodnot označila místa, kde se s největší pravděpodobností číslice budou vyskytovat. Správně určila hodnoty číslic 5 a 7 i s jejich pozicí, dvojku a devítku neidentifikovala vůbec. Zajímavé výsledky můžou nastat, pokud namísto fotky použijeme video. Chybné lokality by v několika snímcích neexistovaly a vykrystalizovaly by tak správně určené hodnoty a pozice (v našem případě SPZ Sanitky).

5 Závěr

Celou práci jsem pojal jako soubor principů, metod a myšlenek z oblastí zpracování obrazu a umělé inteligence. Obě oblasti přispívají svým dílem. I když přínos umělé inteligence je větší, bez pomoci a ulehčení se neobejde. Potvrdil jsem si tak fakt, že nahlížet na problém pouze z jedné strany vede ke stagnaci, neschopnosti se dále rozvíjet a hlavně ke špatným výsledkům.

Program jsem pro názornost implementoval v programovacím jazyce Java. Tato skutečnost se projevila jako nevhodná při testování větších datových sad (stovky neuronů ve skrytých vrstvách), kdy program padal. I přesto se mi většina testů podařila a potvrdila teoretické předpoklady. Zvolená metoda Back-propagation se ukázala jako vhodná k řešení mnoha problémů při vyhledávání číslic. Nedostatky, ať už jde o neschopnost rozpoznat posunuté obrázky, pootočené obrázky apod., jsem řešil implementováním jednoduchých základních algoritmů a podpůrných konstant. Mám na mysli metody typu prahování, rychlosti učení, přesnosti apod.

Možnosti dalšího vývoje se nabízí z obou oblastí. Je možné aplikovat Houghovu metodu pro lepší vyhledávání přímek na papíře a usnadnit tak rotaci pro nakloněné obrázky, adaptivní prahování pro špatně naskenované texty na papíře, použít jiný druh neuronové sítě či vyzkoušet schopnost back-propagation zpracovávat data v reálném čase ke zpracování videa.

Literatura

[1] FAHEY C.: Neuronové sítě s učením o zaostalých chyba množení. [online], Naposledy změněno 1:19, 3. října 2008, [cit 11. 5. 2009]. URL http://www.colinfahey.com/neural_network_with_back_propagation_learning/neural_network_with_back_propagation_learning_cs.html

[2] SEMOTÁN J.: Neuron živého organismu. [online], Naposledy změněno 15:44, 13. března 2003, [cit 14. 5. 2009].
URL http://neuron.felk.cvut.cz/courseware/data/chapter/36nan007/s01.html

[3] MARTÍNEK L., MIŠKA P.: Formální neuron, perceptron a ADALINE - aktivní fáze, adaptace. [online], Naposledy změněno 15:45, 13. března 2003, [cit 11. 5. 2009].
URL http://neuron.felk.cvut.cz/courseware/data/chapter/36nan009/s03.html

[4] Hyperbolický tangens - Wikipedie, otevřená encyklopedie. [online], Naposledy změněno 18:25, 20. prosince 2008, [cit 14. 5. 2009].
URL http://cs.wikipedia.org/wiki/Hyperbolick%C3%BD_tangens

[5] KUČERA J.: Historie neuronových počítačů a sítí. [online], Naposledy změněno 20:14, 17. května 2000, [cit 14. 5. 2009].
URL http://www.fi.muni.cz/usr/jkucera/pv109/2000/xneudert.html

[6] MYSLÍK V.: Neuronové sítě. [online], Naposledy změněno 16:46, 19. října 1999, [cit 14. 5. 2009].
URL http://aldebaran.feld.cvut.cz/~xmyslik/www/neural.html

[7] 36NAN Neuronové sítě a neuropočítače – Wikia education [online], Naposledy změněno 16:44, 5. února 2008, [cit 14. 5. 2009]. URL http://vzdelani.wikia.com/wiki/36NAN_Neuronov%C3%A9_s%C3%ADt%C4%9B_a_neuropo%C4%8D%C3%ADta%C4%8De

[8] Neuronové sítě – referát předmětu Praktikum z programování MFF CUNI. [online], Naposledy změněno 22:07, 26. července 2005, [cit 14. 5. 2009].
URL http://cgg.mff.cuni.cz/~pepca/prg022/mucha/

[9] The Back Propagation Algorithm - The Robert Gordon university. [online], Naposledy změněno 16:22, 6. února 2004, [cit 14. 5. 2009].
URL http://www.rgu.ac.uk/files/chapter3%20-%20bp.pdf

[10] Neuronová síť - Wikipedie, otevřená encyklopedie. [online], Naposledy změněno 14:34, 17. května 2009, [cit 14. 5. 2009].
URL http://cs.wikipedia.org/wiki/Neuronov%C3%A1_s%C3%AD%C5%A5

[11] Luminosity - Wikipedia, the free encyclopedia. [online], Naposledy změněno 5:21, 3. května 2009, [cit 14. 5. 2009].
URL http://en.wikipedia.org/wiki/Luminosity

[12] Rotation representation (mathematics) - Wikipedia, the free encyclopedia. [online], Naposledy změněno 19:56, 7. května 2009, [cit 14. 5. 2009].
URL http://en.wikipedia.org/wiki/Rotation_representation_(mathematics)

Seznam obrázků

Seznam tabulek

Seznam příloh

Příloha 1. CD

Obsahuje:

- Text práce
- Zdrojové kódy
- Spustitelný program
- Manuál programu
- Uložené datové sady
- Uložené neuronové sítě